UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/781,307	02/18/2004	John G. Beltran	G&C 30566.296-US-U1	4847

55895 7590 09/18/2008

GATES & COOPER LLP
HOWARD HUGHES CENTER
6701 CENTER DRIVE WEST, SUITE 1050
LOS ANGELES, CA 90045

EXAMINER	
ABDUL-ALI, OMAR R	

ART UNIT	PAPER NUMBER
2178	

MAIL DATE	DELIVERY MODE
09/18/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/781,307
Filing Date: February 18, 2004
Appellant(s): BELTRAN ET AL.

Jason S. Feldmar
Reg. No.:39,187
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 6/24/2008 appealing from the Office action mailed 1/24/2008.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-18 remain rejected under 35 U.S.C. 103(a) as being unpatentable over Hirsch (US 6,915,301).

Claim 1: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:

a. receiving a reference to an object instance having a dynamic property that is created at runtime for the object instance on a per-instance basis and is not stored with the object (column 11, lines 37-58);

Hirsch discloses retrieving reference to a property source instance (data source) from an association between the object and the property source instance (binding),

wherein the property source instance creates and supplies to dynamic property (column 4, lines 11-34/column 11, lines 37-59), but does not explicitly disclose creating and supplying an initial value for the dynamic property for/to the object instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene. Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.

c. providing the reference to the object and the reference to the property source instance to a control which is configured to: (i) retrieve the dynamic property from the property source and (ii) display the property (object inspector window) in a user interface (column 11, 37-67).

Claim 2: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 1 above, further comprising:

a. the dynamic property is provided by an application that is extending an object property set of the object (column 12, lines 41-58).

Claim 3: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 1 above, further comprising:

a. reference to the property source instance is retrieved from a mapping (binding) of property source instances to object class (column 3, lines 8-27/column 12, lines 41-58).

Claim 4: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 1 above, further comprising:

a. the control is further configured to: retrieve the standard properties for the object; and display the standard properties (column 11, lines 52-67).

Claim 5: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:

a. an object instance having a dynamic property that is created at runtime for the object instance on a per-instance basis and is not stored with the object (column 11, lines 37-59);

Hirsch discloses a property source instance wherein the property source instance creates and supplies the dynamic property (column 11, lines 37-59) but does not explicitly disclose creating and supplying an initial value for the dynamic property for/to the object instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene (column 4, lines 11-34/column 11, lines 37-59). Furthermore, providing initial values

was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.

c. an association between the object and the property source instance (column 3, lines 8-27);

d. a host configured to: (i) retrieve a reference to the object instance; (ii) retrieve a reference to the property source; (iii) provide the reference to the object and the reference to the property source to a control which is configured to (1) retrieve the dynamic property from the property source and (2) display the property in a user interface (column 11, lines 37-67).

Claim 6: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 5 above, further comprising:

a. the dynamic property is provided by an application that is extending an object property set of the object (column 12, lines 41-58).

Claim 7: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 5 above, further comprising:

a. a mapping of property source instances to object classes, wherein the host is configured to retrieve the reference to the property source instance from the mapping (column 12, lines 41-58).

Claim 8: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 5 above, further comprising:

a. the control is further configured to: retrieve the standard properties for the object; and display the standard properties (column 11,lines 52-67).

Claim 9: Hirsch discloses a method and computer system for dynamic properties of software objects comprising receiving a first object having a first property wherein the first object provides a control (calendar control) that defines a first user interface for displaying and editing the first property, but does not explicitly disclose the control is an ActiveX control. However, Hirsch does disclose supporting ActiveX controls (column 12, lines 8-31). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to provide an ActiveX control to display and edit the first property. One would have been motivated to provide an ActiveX control to display and edit the first property in order to make the design more efficient.

b. creating a list of one or more object properties to be displayed, wherein the list includes the first property (column 12, lines 8-31);

c. instantiating the custom ActiveX control (column 12, lines 8-31);

d. displaying the object properties in the list, wherein the display of the first property comprises the first user interface defined by the instantiated ActiveX control, wherein the property may be edited through the first user interface (column 12, lines 8-31).

Claim 10: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 9 above, further comprising:

a. instantiating one or more stock ActiveX controls that define one or more additional user interfaces for displaying and editing remaining object properties in the list, wherein the stock ActiveX controls are not provided by any object containing one or more of the remaining object properties (column 12, lines 8-31).

Claim 11: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 10 above, further comprising:

a. the first user interface and additional user interfaces are displayed in a single dialog box (column 12, lines 8-31).

Claim 12: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 9 above, further comprising:

a. an application programming interface provides the ability to push the first object to a second object for display (column 12,lines 41-58).

Claim 13: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:

a. one or more objects, wherein each object has one or more object properties (column 11, lines 37-59);

b. a property inspector configured to (i) interrogate the one or more objects to discover one or more object properties to be displayed; (ii) create a list of the one or more object properties to be displayed; (iii) instantiate and host one or more property editors (column 11, lines 37-59/column 12, lines 1-21);

Hirsch discloses one or more property editors wherein: (i) one of the property editors comprises a custom control specified by one of the objects, but does not explicitly disclose the control is an ActiveX control. However, Hirsch does disclose supporting ActiveX controls (column 12, lines 8-31). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to provide an ActiveX control specified by one of the objects. One would have been motivated to provide an ActiveX control specified by one of the objects in order to make the design more efficient.

Hirsch discloses (ii) the custom ActiveX control defines a custom graphical user interface for displaying and editing one of the object properties (column 12, lines 8-31).

Claim 14: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 13 above, further comprising:

a. one of the property editors is comprised of a stock ActiveX control that defines an additional user interface for displaying and editing one or more additional properties in the list, wherein the stock ActiveX control is not provided by one of the objects that contains the one or more additional properties (column 12, lines 8-31).

Claim 15: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 14 above, further comprising:

a. the custom graphical user interface and additional user interfaces are displayed in a single dialog box (column 11, lines 37-59/column 12, lines 1-21).

Claim 16: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 13 above, further comprising:

a. an application programming interface configured to provide the ability to push the one or more objects to the property inspector for display (column 12, lines 41-58).

Claim 17: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:

a. an object instance of a class, wherein (i) an initial value for one or more static properties of the class are assigned at run time; and (ii) the object instance has a dynamic property that is generated by a property source instance at runtime for the object instance on a per-instance basis and are not stored with the object (column 12, lines 20-58/column 11, lines 37-59). Hirsch does not explicitly disclose an initial value is

generated and supplied by a property source instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene. Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.

b. an association between either: (i) the object instance and the property source instance; (ii) the class and the property source instance (column 11, lines 37-61)

c. a user interface component that displays a collection of properties of the object instance including the one or more static properties and the dynamic property on a display device(column 11, lines 37-61), wherein the user interface component is configured to:

(i) retrieve a reference to the object instance (column 11, lines 49-63);

(ii) retrieve the one or more static properties from the object instance (column 11, lines 49-63);

(iii) access the association to determine the property source instance associated with the object instance (column 11, lines 37-61);

(iv) call a method of the determined property source instance with the reference to the associated object instance (column 11, lines 37-61);

(v) receive the dynamic property, from the property source instance, wherein the property source instance dynamically generated the dynamic property (column 11, lines 49-63). Hirsch does not explicitly disclose an initial value is generated and supplied by a property source instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene. Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.

(vi) display the static property and the dynamic property on the display device (column 11, lines 49-63).

Claim 18: Hirsch discloses a method and computer system for dynamic properties of software objects as in Claim 9 above, further comprising custom controls are provided on a per-instance basins, but does not explicitly disclose the controls are ActiveX controls. However, Hirsch does disclose supporting ActiveX controls (column 12, lines 8-31). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to provide an ActiveX control on a per-property basis.

One would have been motivated to provide an ActiveX control on a per-property basis in order to make the design more efficient.

b. two or more object properties are in the properties list (column 12, lines 8-31);

c. the two or more object properties are displayed in a list, wherein each of the two or more object properties are displayed using user interfaces defined by the custom ActiveX controls (column 12, lines 8-31).

(10) Response to Argument

Appellant argues that Hirsch does not explicitly disclose all of the claim limitations of Claims 1-18.

Independent Claims 1, 5, and 17

Argument 1: "Hirsch completely and entirely fails to teach, describe, or suggest, explicitly of implicitly the creation of the creation of a property source instance for a property of a separate instance."

Appellants do not assert that the claims provide for creating a property source instance

Argument 2: Hirsch further fails to teach, describe, or suggest the creation of a value for such a dynamic instance. Hirsch's dynamic property merely refers to properties 112 and 122 that are dynamic in the sense that they are evaluated at run time and are static

constants. Thus, Hirsch's dynamic properties already exist for an object but are merely evaluated at run time (page 14). Merely evaluating an expression for a property does not provide for nor does it create the property itself. Such a teaching neither teacher nor discloses, explicitly of implicitly the claimed property and initial value for the property (page 18).

Argument 3: Nowhere in Hirsch is there even a remote reference to a control receiving both a reference to an object instance and a property source instance, and then retrieving the dynamic property from the property source instance and displaying such a property.

The Examiner respectfully disagrees.

Response to Argument 1: Appellant argues that Hirsch completely and entirely fails to teach, describe, or suggest, explicitly of implicitly the creation of a property source instance for a property of a separate instance. As noted previously in the Advisory Action, the claim language of Claim 1 does not reflect the creation of a property source instance for a property of a separate instance. The only creation step reflected in the claim language of Claims 1, 5, and 17, is the creation of a dynamic property and an initial value for the dynamic property. The property source instance creates and supplies the dynamic property and an initial value for the property for/to the object instance (Claims text).

Response to Argument 2: Hirsch further fails to teach, describe, or suggest the creation of a value for such a dynamic instance. Hirsch's dynamic property merely refers to properties 112 and 122 that are dynamic in the sense that they are evaluated at run time and are static constants. Thus, Hirsch's dynamic properties already exist for an object but are merely evaluated at run time (page 14). Merely evaluating an expression for a property does not provide for nor does it create the property itself. Such a teaching neither teacher nor discloses, explicitly of implicitly the claimed property and initial value for the property (page 18). Appellants question where Hirsch provides for dynamically creating both a property and a value for that property (as claimed). All of the actions ignore the limitation relating to creating the property itself. Nowhere in Hirsch is there even a remote reference to the creation of such a property (page 19).

In response to Applicant's assertion, the creation of a value for a property source instance is not supported by the claim language of Claims 1, 5, and 17 based on the above interpretation. The property source instance creates the dynamic property and a value for the dynamic property (Claims text). Hirsch discloses object properties consist of named attributes that define an object's appearance in terms of a functional expression. The values of these properties are set at runtime based on the calculated result of the expressions. The creation of a value at runtime based on a calculated expression is dynamic and is created on a per instance basis. Data sources (property sources) are named SQL queries that are executed and used in a layout. When used in

a layout, each row resulting from the query is transformed into a graphical representation represented by a data element (column 5, lines 5-19). Data elements consist of objects (column 4, lines 44-51; object instance), whose properties contain values that are set at runtime based on calculated expressions, making them dynamic properties. This teaching provides reasonable suggestion that dynamic values are created for dynamic properties in Hirsch. Hirsch further supports additional dynamic properties (Fig. 2, '112', '122') which are evaluated at runtime.

Response to Argument 3: Applicant argues, "Nowhere in Hirsch is there even a remote reference to a control receiving both a reference to an object instance and a property source instance, and then retrieving the dynamic property from the property source instance and displaying such a property (page 15). Appellants further question where Hirsch describes the ability to retrieve a dynamically created property from a property source instance and displaying the property (page 17)."

The Examiner respectfully disagrees. Hirsch discloses an object inspector window displays the object, the objects properties, and receives the object properties from the data source. The execution of a data source results in the creation of a data element, which includes objects whose properties contain values that are set at runtime based on calculated expressions. The retrieval of this information by the object inspector window to display the information provides the teaching of a reference between the property source (data source) and the object instances (objects). The object inspector window displays object attributes that define an object's appearance in

terms of a functional expression, and also displays additional dynamic properties which are evaluated at run time (column 11, lines 37-61).

Dependent Claims 2 and 6

Argument 4: Such an object model does not remotely reflect or relate to an application that is extending an object property set of an object.

Response to Argument 4: Hirsch discloses a virtual world associated with an application that is built using building blocks such as scenes, data sources, global parameters, and resources (Abstract). Data sources provide objects through the creation of data elements (column 4, lines 43-51). An objects property set in Hirsch includes named attributes that define an object's appearance in terms of a functional expression. Their values are set at runtime based on the calculated result of the expressions. The property set is extended through the use of calculated expressions, in the sense that different expressions would provide various calculated results.

Dependent Claims 3 and 7

Argument 5: Nowhere in the text is there even a remote hint at a mapping between property source instances and object class. [Hirsch's] teaching does not relate to, explicitly or implicitly, the ability to map a property source instance to an object class

and retrieving a reference to a property source instance from such a mapping as claimed.

Response to argument 5: Hirsch discloses dynamic object properties are bound to data, and the bindings are applied to an entire class of object rather than simply to a single object (column 3, lines 11-23). Data sources (property source instances) are bound to objects through one or more data values (column 6, lines 60-67). Therefore, property source instances are mapped to an object class in Hirsch. The reference is retrieved through the use of the object inspector window, which displays an object's properties.

Dependent Claims 4 and 8

Argument 6: The claims provide for different types of properties – standard properties and the dynamically created properties – the action fails to acknowledge such differences and Hirsch fails to teach such differences.

Response to Argument 6: Hirsch discloses static properties (standard properties) and dynamic properties may be displayed by the object inspector window (column 11, lines 29-60).

Independent Claims 9 and 13

Argument 7: Nowhere in Hirsch is there any description that the object itself provides the custom control to display the properties of the object. In contrast to Hirsch's teaching, the presently claimed invention provides that the object provides a custom control that defines a user interface for displaying and editing one of its own properties. Such a capability is wholly and completely missing, both explicitly and implicitly from Hirsch.

Argument 8: Hirsch does not even remotely describe that the control is a custom control. Hirsch fails to provide such a custom control that defines a user interface for displaying and editing the property. There is not even a remote hint at defining a custom control.

The Examiner respectfully disagrees.

Response to Argument 7: Hirsch discloses an object inspector window that displays object properties. When the properties tab is selected, the user may enter property values that may be constants or calculated values containing functions of parameters of column names from a data source. If a property value is of an enumerated type (a value that can only be set to a finite number of values), a drop down combo-box may be displayed listing the legal values for the property. When a property is a date or time, a calendar control (custom control) may be displayed beneath the property value when

the field is active. This is a teaching of a user interface for displaying and editing properties (selecting dates or times through a calendar control). Based on the broadest, reasonable interpretation, the objects provide these custom controls. An object that does not include a date or time property would not provide the calendar control when inspected by the object inspector window, giving the objects the ability to "provide" custom controls based on which object is being inspected.

Response to Argument 8: Hirsch discloses a calendar control *may* be displayed beneath the property value when the field is active (emphasis added). Hirsch provided earlier teaching that the user may enter values that are static constants in the column to define an object's property values, so instead of using a calendar control, a user may simply enter the value. Hirsch's controls can be reasonably interpreted as custom, based on the fact that they are used according to the specification of Hirsch's design.

Dependent Claims 10 and 14

Argument 9: The last part of this claim clearly differentiates Hirsch from the present invention. The object inspector cannot provide any control to display other properties. Hirsch's object inspector could not provide both types of Active X controls.

Response to argument 9: The object inspector provides a 2 column entry form, where a user may use a drop-down combo box (stock control) or a calendar control (custom

control) to edit the value of an object. The drop-down combobox is instantiated to provide an additional user interface for displaying and editing remaining object properties in the list. For example, property "Font name" would provide a drop-down combobox with the values: "Arial, and "Courier". Property "Color" would provide a drop-down combobox with the values: "Red" and "Blue". These controls are not provided by any object containing one or more of the remaining object properties, in the sense that a objects that only contain date and time properties would not provide the stock combobox controls.

Dependent Claims 12 and 16

Argument 10: The use of a push based model or the ability to push a first object to a second object for display is not even remotely hinted at anywhere in the cited text or remainder of Hirsch.

Response to Argument 10: Hirsch discloses the ability to push the first object to a second object for display. Objects (first object) and their properties are displayed in a property inspector window (second object). The object inspector is displayed when a scene editor or data element editor window is active in design mode (column 11, lines 49-61/column 12, lines 1-8). The objects are pushed to the object inspector window in order to examine and edit their properties. Hirsch describes a set of functions (application programming interface) such as VcPropertyBag, which provides an

interface for accessing and manipulating properties and their values (column 12, lines 41-67).

Dependent Claim 18

Argument 11: To be more efficient, a single ActiveX control used to display all properties would be more efficient than using an ActiveX control on a per-property basis. Thus the Examiner's relied upon motivation would actually serve to teach away from the present invention.

Argument 12: Hirsch fails to even remotely describe using ActiveX controls on a per property basis as claimed:

Response to Argument 11: By providing controls on a per property basis, operator efficiency would be increased. For instance, by providing a calendar control for a property that is based on dates, a user would be able to quickly enter a date through the control, as opposed to entering the day, month, and year individually.

Response to Argument 12: Hirsch discloses controls that are provided on a per property basis. If a property value is of an enumerated type (a value that can only be set to a finite number of values), a drop down combo-box may be displayed listing the

legal values for the property. When a property is a date or time, a calendar control (custom control) may be displayed beneath the property value when the field is active.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Omar Abdul-Ali

9/14/2008

/OAA/

Conferees:

/Stephen S. Hong/

Supervisory Patent Examiner, Art Unit 2178

/DENNIS-DOON CHOW/
Supervisory Patent Examiner, Art Unit 2173